Exercise 9.1.

transitive graph $\Longleftrightarrow$ disjoint union of complete graphs

i) Prove "$\Longleftarrow$".

- Let $G = (V, E)$ be a disjoint union of complete graphs $G_1 = (V_1, E_1), \dots, G_k = (V_k, E_k)$.

- Let $\{(u,v),(v,w)\} \in E$. Since G is a disjoint union, $\exists i \in [1,k]$ such that $v \in V_i$ and $\{(u,v),(v,w)\} \subseteq E_i$.

- We know $E_i = \{(x,y) \mid x,y \in V_i, x \neq y\}$ and $\{u,v,w\} \subseteq V_i$, hence $(u,w) \in E_i$.

ii) Prove "$\Longrightarrow$".

Let $G = (V, E)$ be a transitive graph. Decompose G into its connected components $G_1 = (V_1, E_1), \dots, G_k = (V_k, E_k)$. G is the disjoint union of its connected components.

We only need to prove that each connect component is complete. Choose an arbitrary connected component $G_i = (V_i, E_i)$. Choose $u, v \in V_i$, $u \neq v$. There exists a path $u = w_1 \to w_2 \to \dots \to w_p = v$ from u to v in G_i.

Show by induction on $j \in \{2, \dots, p\}$: $P(j)$: "the edge (u, w_j) is in E_i". (Here we prove for each path starting from u that there is an edge between u and every vertex

that there is an edge between v and every vertex w_j in path. And since there is a path between every vertex inside a connected component this proof implies that the connected graph is a complete graph)

Base Case ($j=2$): As the path is $v=w_1 \to w_2 \to \dots \to w_p$ we obviously have the edge $(v=w_1, w_2)$ in E_i.

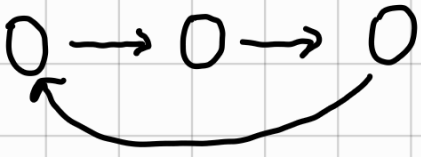
Induction Step ($j \to j+1$): Let $j \in [2, \dots, p-1]$ such that $P(j)$ holds. So, we have an edge $(v, w_j) \in E_i$. We also have the edge $(w_j, w_{j+1}) \in E_i$. By transitivity we get $(v, w_{j+1}) \in E$. This shows $P(j+1)$.

Exercise 9.2

a) $\forall v \in V \ \deg_{in}(v)$ is even $\implies |E|$ is even

True. Because $\sum_{v \in V} \deg_{in}(v) = \sum_{v \in V} \deg_{out}(v) = E$

b) No. It would have held if G was a DAG.



is a counter example.

c) It is a DAG, so topological sort exists.

Exercise 9.3

e) Here we don't know which v is chosen. It can be any vertex. In worst-case v is the vertex with most degree such that v is in the last entry of its adjacency list.

In this case, required time for edge deletion is $\deg(v) = \max_{w:\{v,w\}\in E} \deg(w)$. In the case of improved adjacency list we can just use the pointer to access the entry.

f) Traverse adjacency lists of u and v at the same time. If we are finished with one of the lists and couldn't find the edge, insert the edge immediately to both lists. This requires $\Theta(1 + \min_{x\in\{u,v\}}(\deg x))$.

g) Copy the complete matrix, leave out the row and column corresponding to v. This takes time $\Theta(n^2)$.

If we don't delete v, we can do everything in $\Theta(n)$.

Exercise 9.4.

Source has $\deg_{in} = 0$.
Sink has $\deg_{out} = 0$.

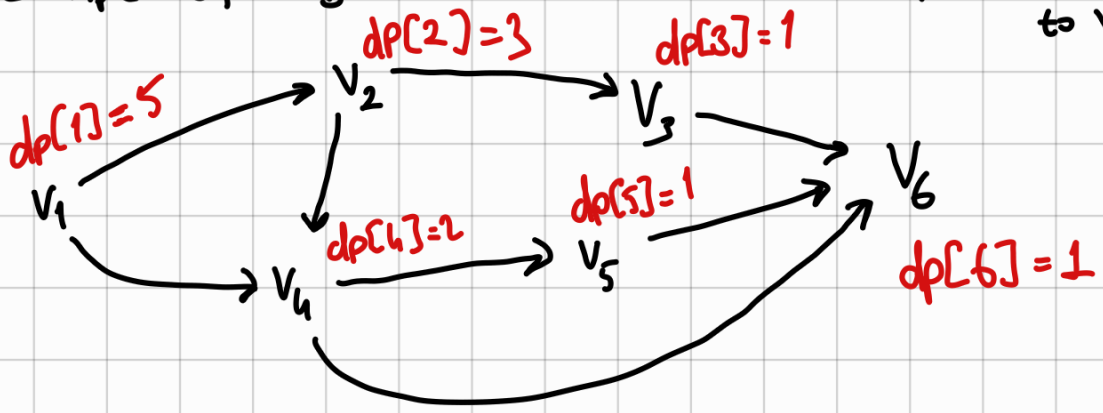
a) Let G' be the graph without v_1 and v_n. There exists a topological sorting of G' since it is a DAG. Add v_1 to the beginning and v_n to the end of sorting, we get a topological sorting for G.

b) For any edge (v_i, v_j) we have $i<j$.

For every v_1-v_n-path $P: v_1 = v_{i_0}, v_{i_1}, v_{i_2}, \dots, v_{i_\ell} = v_n$ then $(v_{i_0}, v_{i_1}), (v_{i_1}, v_{i_2}), \dots, (v_{i_{\ell-1}}, v_{i_\ell})$ are edges of G and thus

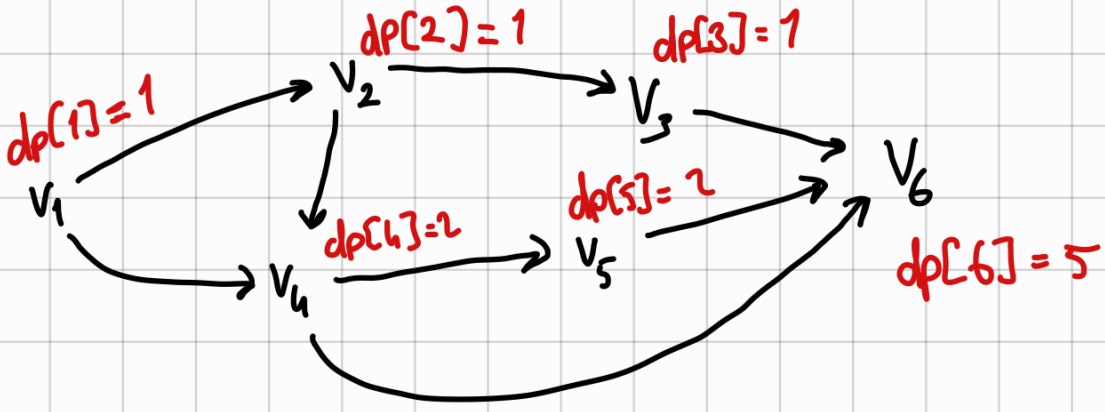
$i_0 < i_1 < \ldots < i_\ell$.

c) Example of DP-Entries. (DP[i]: number of paths in G from v_i to v_n)

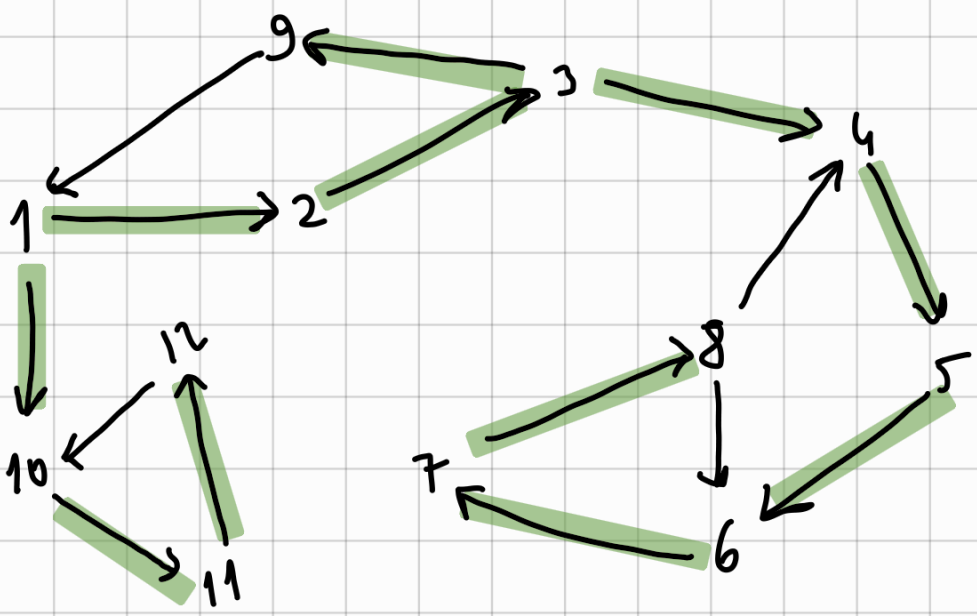


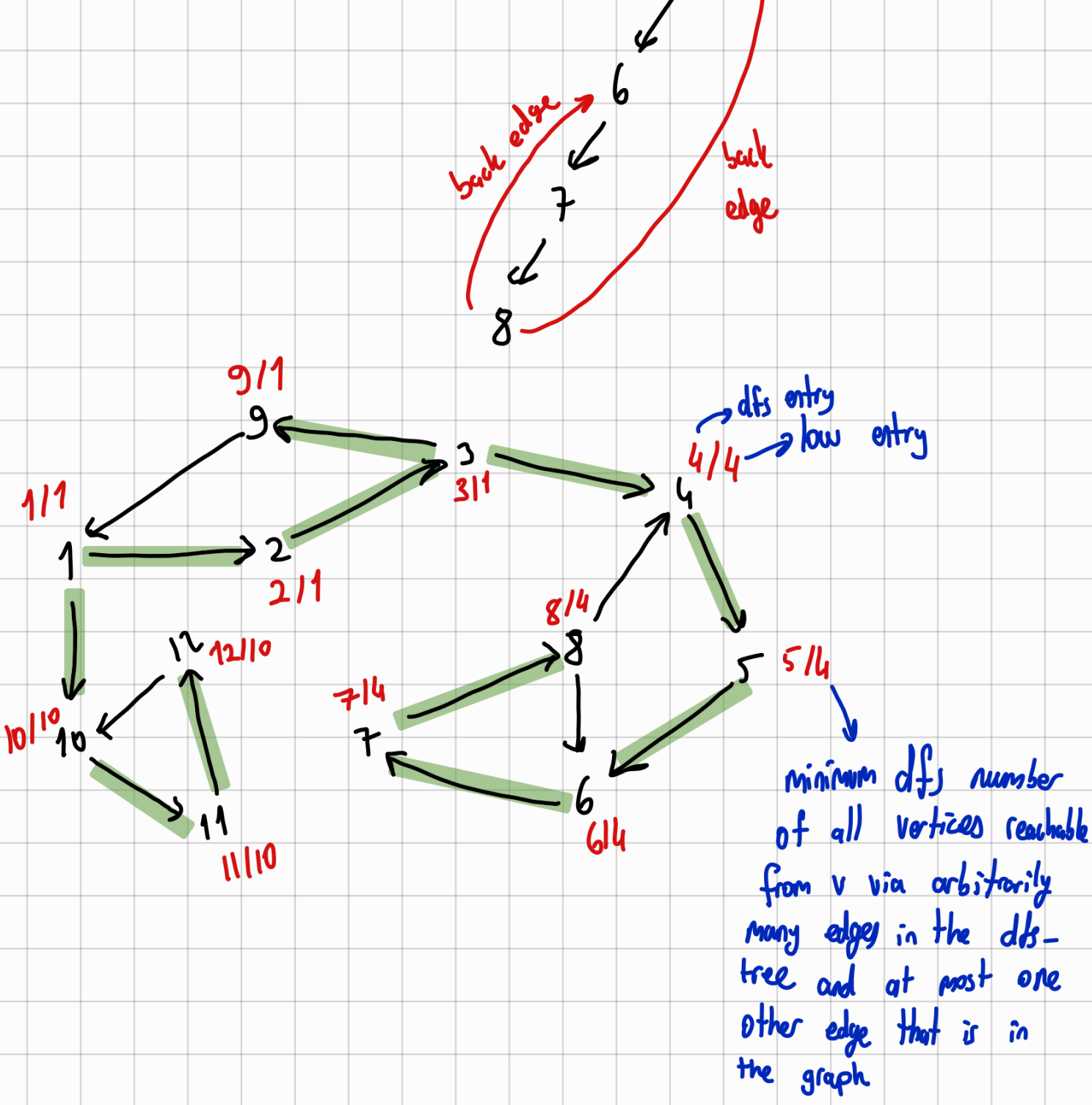
$$DP[i] = \sum_{j \in \{i+1,\ldots,n\}:\ (v_i,v_j)\in E} DP[j]$$

Alternative: (DP[i]: number of paths in G from v_1 to v_i)



Additional





articulation point iff • v = source and has degree ≥ 2

• v ≠ source $\exists w \in V$ with $\{v,w\} \in E(T)$: $low[w] \geq dfs[v]$

↳ a.p. if one of their children in the search tree has a low number at least as big as their dfs number

Bridge iff. $(u,v) \in E(T) \wedge low(v) > dfs[u]$ or
$(v,u) \in E(T) \wedge low(u) > dfs[v]$